A Web Application for Geolocalized Signs in Synthesized Swiss German Sign Language

Jancso, Anna ; Rao, Xi ; Graën, Johannes ; Ebling, Sarah

DOI: https://doi.org/10.1007/978-3-319-41267-2_62

A Web Application for Geolocalized Signs in Synthesized Swiss German Sign Language

Anna Jancso, Xi Rao, Johannes Graën, and Sarah Ebling(✉)

Institute of Computational Linguistics, University of Zurich, Zurich, Switzerland
{anna.jancso,xi.rao}@uzh.ch, {graen,ebling}@cl.uzh.ch

Abstract. In this paper, we report on the development of a web application that displays Swiss German Sign Language (DSGS) signs for places with train stations in Switzerland in synthesized form, i.e., by means of a signing avatar. Ours is the first platform to make DSGS place name signs accessible in geolocalized form, i.e., by linking them to a map, and to use synthesized signing. The latter mode of display is advantageous over videos of human signers, since place name signs for any sign language are subject to language change. Our web application targets both deaf and hearing DSGS users. The underlying programming code is freely available. The application can be extended to display any kind of geolocalized data in any sign language.

Keywords: Avatar · Sign language · Geolocalized signs · Web platform

1 Introduction

Swiss German Sign Language (*Deutschschweizerische Gebärdensprache*, DSGS) is the sign language of the German-speaking area of Switzerland. In any sign language, names for geographical locations (such as city or country names) can be signed in different ways, of which the most common are [4]:

- Using a conventionalized *place name sign*, e.g., the sign for GENF ('GENEVA')[1] in DSGS as shown in Fig. 1;
- *Fingerspelling* the place name, i.e., using dedicated signs for the letters of the place name in the surrounding spoken language[2] (the *finger alphabet* of DSGS is shown in Fig. 2); or
- Pointing in the relative compass direction of a place and articulating with the mouth (i.e., *mouthing*) the spoken language place name.

Which of the three above devices is chosen depends, among other factors, on the prominence of the geographical location and the origin of the signer.

[1] "GENF" is an example of a *sign language gloss*, a label for the meaning of a sign. Glosses are commonly written in all caps.

[2] A *spoken language* is a language that is not signed, whether it is represented as speech or text.

K. Miesenberger et al. (Eds.): ICCHP 2016, Part II, LNCS 9759, pp. 438–445, 2016.
DOI: 10.1007/978-3-319-41267-2_62

Fig. 1. Sign GENF ('GENEVA') in DSGS [1]

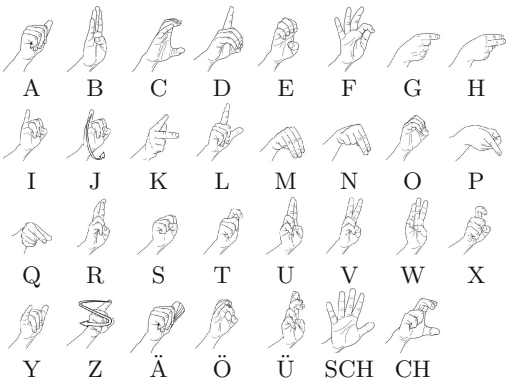

Fig. 2. Finger alphabet of DSGS [1]

For example, a DSGS signer from Zurich may not know the sign for the place Muttenz, a town outside of Basel. Hence, he is likely to sign the place name using fingerspelling or pointing and mouthing. However, using conventionalized signs for place names where possible is desirable. This makes an inventory of place name signs an indispensable resource for any sign language. Often, these inventories form part of a general sign language lexicon, as is the case for DSGS.[3]

In this paper, we report on the development of a web application that displays DSGS signs for places with train stations in Switzerland in synthesized form, i.e., by means of a *signing avatar*. Ours is the first platform to make DSGS place name signs accessible in geolocalized form, i.e., by linking them to a map, and to use synthesized signing. The latter mode of display is advantageous over videos of human signers, since place name signs for any sign language are subject to language change. As an example, the sign for the place *Winterthur* in DSGS has recently changed.

[3] http://signsuisse.sgb-fss.ch/ (last accessed: January 14, 2016).

Our web application targets both Deaf[4] and hearing DSGS users and is accessible at http://pub.cl.uzh.ch/purl/geosign. The source code of our web application (including database schema, SiGML codes, train station names and their coordinates, web frontend and backend scripts) is available at the same address.

2 Synthesized Signing

The Java Avatar Signing (JASigning) system [3] was used to synthesize the DSGS place names. Its main release is freely available for research purposes.[5] JASigning accepts input in the form of Signing Gesture Markup Language (SiGML) [2], an XML version of the form-based sign language notation system HamNoSys [6]. Figure 3 shows the SiGML code for GENF (cf. Fig. 1) in DSGS.

```
<sigml>
  <hamgestural_sign gloss="GENF">
    <sign_nonmanual>
    </sign_nonmanual>
    <sign_manual>
      <handconfig ceeopening="slack" handshape="cee12" mainbend="bent"
↪   second_handshape="finger2" second_thumbpos="out"/>
      <handconfig extfidir="ul"/>
      <handconfig palmor="l"/>
      <location_bodyarm contact="touch" location="chin"
↪   side="right_beside">
        <location_hand digits="1" location="nail" side="palmar"/>
      </location_bodyarm>
      <par_motion>
        <directedmotion direction="do"/>
        <tgt_motion>
          <changeposture/>
          <handconfig handshape="pinch12" mainbend="bent"/>
          <handconfig extfidir="uol" palmor="ul"/>
        </tgt_motion>
      </par_motion>
    </sign_manual>
  </hamgestural_sign>
</sigml>
```

Fig. 3. SiGML code for the place name sign GENF ('GENEVA') in DSGS

[4] It is a widely recognized convention to use the upper-cased word *Deaf* for describing members of the linguistic community of sign language users [5].

[5] http://vh.cmp.uea.ac.uk/index.php/JASigning (last accessed: January 14, 2016).

The names and positions (longitude, latitude) of all train stations of the Swiss Federal Railways (*Schweizerische Bundesbahnen*, SBB) are available online.[6] From this list, we extracted 1829 stations that we wanted to display in synthesized DSGS. For 232 of these stations, conventionalized DSGS signs (cf. Sect. 1) could be identified. For the remaining stations, the names are signed using fingerspelling (cf. Sect. 1).

The SiGML codes for fingerspelling signs were generated by first normalizing and segmenting the station names and finally concatenating the corresponding partial SiGML codes. Normalizing involved handling diacritics in French and Italian place names (e.g., converting *à* to *a*). For segmentation, recall from Fig. 2 that the DSGS finger alphabet features 31 signs. Segmenting the station names according to fingerspelling signs included identifying the letter combinations *sch* and *ch* as one segment. Successive vowels and consonants were also combined into a single segment: In DSGS, double vowels are signed by adding a horizontal movement (a *glide*) to the sign for the vowel, and for double consonants, a small stamping movement is appended after each consonant. Overall, this resulted in 56 partial SiGML codes.

3 Implementation

The SiGML documents for those train stations that have a place name sign (cf. Fig. 3) and the partial SiGML codes for the finger alphabet (cf. Fig. 2) are stored in a database, together with the station names, the coordinates of the stations (longitude and latitude), and a *category* parameter assigning every station to one of five categories (cf. Sect. 3.4).

A stored procedure returns valid SiGML code for any given word, defaulting to a fingerspelled realization if no SiGML code for a lexical sign is found.

3.1 Architecture

We chose Python as a server-side language for retrieving data from the database and delivering it to the web client. The web client can make two different requests: The first extracts all the information regarding the train stations needed for creating markers on a map (cf. Sect. 3.2) and is raised just before the initialization of the map. The second request is executed when the user clicks on a marker or issues a query for a specific train station. As a response, the corresponding SiGML code is returned and passed to the avatar. Both requests are asynchronously handled via *Ajax* on the client side to avoid "freezing" and reloading of the entire web page.

[6] http://www.bav.admin.ch/dokumentation/publikationen/00475/01497/index.html (last accessed: January 14, 2016).

```
<?xml version="1.0" encoding="UTF-8"?>
<sigml>
<player_settings>
  <camera_location cx="0.00" cy="0.40"
        r="2.20" theta="0.0" phi="10.0" fov="30.0"/>
  <ambient_motions body="ON" head="ON" blink="ON"/>
  <avatar name="bad"/>
</player_settings>

<hamgestural_sign gloss="TAESCH">
  <sign_manual>
    <sign_manual>
      <handconfig bend1="hooked" bend2="round" handshape="finger2"
↪   mainbend="round" thumbpos="out"/>
      <handconfig extfidir="u"/>
      <handconfig palmor="dl"/>
      <location_bodyarm location="shoulders" side="right_at"/>
    </sign_manual>
    <sign_manual>
      <handconfig handshape="fist" second_handshape="fist"
↪   thumbpos="out"/>
      <handconfig extfidir="u"/>
      <handconfig palmor="d"/>
      <location_bodyarm location="shoulders" side="right_at"/>
      <rpt_motion repetition="fromstart">
        <tgt_motion>
          <changeposture/>
          <handconfig handshape="fist"/>
        </tgt_motion>
      </rpt_motion>
    </sign_manual>
    <sign_manual>
      <handconfig handshape="finger2345" thumbpos="out"/>
      <handconfig extfidir="u"/>
      <handconfig palmor="d"/>
      <location_bodyarm location="shoulders" side="right_at"/>
    </sign_manual>
    </sign_manual>
</hamgestural_sign>

<hamgestural_sign>
</hamgestural_sign>
</sigml>
```

Fig. 4. Concatenated SiGML code for the fingerspelling sign TÄSCH in DSGS

3.2 The Map

Our aim was to visualize the train stations on a map by means of markers. Clicking on a marker or searching for a train station triggers a pop-up window containing the avatar, which subsequently signs the train station name in DSGS. We used Google Maps as map material and implemented its API to configure our map (cf. Sect. 3.5).

3.3 Coordinates

In the coordinate data described in Sect. 2, longitude and latitude information was available only in LV03 (*Landesvermessung 1903*) format. Since the Google Maps API only accepts coordinates in the WGS84 (*World Geodetic System 1984*) format, we converted the LV03 coordinates using a Python module.[7]

3.4 Clustering Markers

To avoid overload of markers on the map, the number of markers was set to increase with the zoom levels. The most intuitive criterion for selecting train stations for the different zoom levels is passenger frequency, i.e., the number of passengers getting on/off at a given train station (as obtained from a list provided by the SBB). However, this results in an uneven distribution of markers on the map in some cases: For example, *Zürich Stadelhofen* and *Zürich HB* are both ranked high (9 and 1, respectively) in the list of passenger frequencies. Since they are located close to each other, displaying the markers for both stations is not ideal. We therefore introduced proximity to other stations as an additional criterion. We then defined five categories, imposing a specific threshold value for the distances: *high* (0.15, 185 stations), *high-medium* (0.075, 235 stations), *medium* (0.0375, 434 stations), *medium-low* (0.01875, 473 stations), and *low* (0.009375, 502 stations).

3.5 Zooming

Once the web application is loaded, a map is initialized, centering on Switzerland at zoom level 8.[8] For every train station, a marker is generated by passing its coordinates and name to the constructor of a class called *Marker*. The avatar is integrated into a pop-up window. For this, an object of the class *InfoWindow* is created and associated with a specific marker. An event listener is then added to every marker to ensure that the appropriate info window opens upon clicking on a marker. Initially, i.e., when the page is loaded, only markers belonging to the highest category are shown on the map. Zooming in results in more markers

[7] http://www.swisstopo.admin.ch/internet/swisstopo/en/home/products/software/products/skripts.html (last accessed: January 14, 2016).

[8] Zoom level 0 is the most zoomed out (the entire world is shown), meaning that you need to zoom in eight times to get to zoom level 8 (Switzerland).

being placed on the map, while zooming out removes them again. To implement this behavior, we first added another event handler, which fires as soon as a change in the zoom level is detected. To recognize whether the user zooms in or out, the new zoom level and the previous zoom level are compared. Zoom levels 8 to 12 are then mapped (1) to the category where stations need to be put on the map (zooming in) and (2) to the category where stations need to be removed from the map (zooming out). Based on the old and new zoom levels and the mapping, the algorithm decides on the next action.

3.6 Searching for Stations

As an additional feature, we provide a search mask with autocompletion function.[9] This function renders the query for train stations easier and more precise because it prevents the user from typing and submitting any non-existing name. Since the stations are sorted by rank during retrieval from the database, the ones with higher frequency appear first in the list. Once a station name in the input field is submitted, the application looks up the corresponding marker. If found, the marker is set on the map (provided this has not yet been done), the corresponding SiGML code is retrieved, the info window is opened, and the signing avatar is triggered.

3.7 Embedding the Avatar Applet

HTML files with sample embeddings of the avatar using JavaScript are provided on the JASigning website (cf. Sect. 2). HTML code is passed to the constructor of the *InfoWindow* class. We modified a method in the original JavaScript code so that it now operates on the SiGML code extracted from the database instead of reading the content from a file. The method is invoked by the *Play* button below the avatar. Figure 5 shows a screenshot of our web application.

4 Outlook

DSGS is composed of five dialects that originated in former schools for the Deaf, resulting in Zurich, Berne, Basel, Lucerne, and St Gallen dialects. Currently, our application contains only signs from the Zurich dialect. As a next step, we intend to include all dialect variants of a place name sign. The display of the variants will be similar as that of the *Spread the Sign* web page.[10]

Our database can be extended to not only contain SiGML documents for train station names, but any word/gloss in DSGS, thus providing a resource for web applications based on signing avatars. If interchanged or extended with the finger alphabet SiGML codes of another sign language, the system will support that language instantly.

[9] https://api.jqueryui.com/autocomplete/ (last accessed: January 14, 2016).
[10] https://www.spreadthesign.com/de/map/ (last accessed: January 14, 2016).

Fig. 5. Screenshot of our web application

Acknowledgments. Synthesized signing was produced with the JASigning software developed at the University of East Anglia, UK. We gratefully acknowledge the work of Prof John Glauert, Dr Ralph Elliott, Dr Richard Kennaway, and Mr Vincent Jennings on Animgen. Furthermore, our thanks go to the Bundesamt für Verkehr and Swisstopo, where we obtained the relevant data for the train stations of the SBB and a Python script to convert the coordinates.

References

1. Boyes Braem, P.: A multimedia bilingual database for the lexicon of Swiss German Sign Language. Sign Lang. Linguist. **4**(1/2), 133–143 (2001)
2. Elliott, R., Glauert, J., Kennaway, R., Marshall, I.: The development of language processing support for the ViSiCAST project. In: Proceedings of the 4th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS), pp. 101–108. Arlington, VA (2000)
3. Jennings, V., Elliott, R., Kennaway, R., Glauert, J.: Requirements for a signing avatar. In: Proceedings of the 4th LREC Workshop on the Representation and Processing of Sign Languages, pp. 133–136. La Valetta, Malta (2010)
4. Matthews, P., McKee, R., McKee, D.: Signed languages, linguistic rights and the standardization of geographical names. In: Proceedings of the 23rd International Congress of Onomastic Sciences, pp. 721–732. Toronto, Canada (2008)
5. Morgan, G., Woll, B.: The development of complex sentences in British Sign Language. In: Directions in Sign Language Acquisition: Trends in Language Acquisition Research, pp. 255–276. John Benjamins, Amsterdam, Netherlands (2002)
6. Prillwitz, S., Leven, R., Zienert, H., Hanke, T., Henning, J.: HamNoSys: Version 2.0: An Introductory Guide. Signum, Hamburg (1989)